NPS-AM-13-C10P01R05-034

EXCERPT FROM THE

PROCEEDINGS

OF THE

TENTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM ACQUISITION MANAGEMENT

Software Acquisition Patterns of Failure and How to Recognize Them

Lisa Brownsword, Cecilia Albert, Patrick Place, and David Carney
Carnegie Mellon University

Published April 1, 2013

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
01 APR 2013		**00-00-2013 to 00-00-2013**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Software Acquisition Patterns of Failure and How to Recognize Them	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,Software Engineering Institute (SEI),Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In systems today, software provides substantial portions of capability and performance. In many Department of Defense (DoD) acquisitions, however, software too often is a minor consideration when the early and most constraining decisions about program cost, schedule and behavior are made (i.e., prior to Milestone A). These decisions manifest themselves in the acquisition strategy, the system and software architecture, and ultimately in the deployed system. Based on our experience with large programs, we have identified seven patterns of failure that lead to misalignment between the software architecture and the acquisition strategy, leading to program restarts, cancellations, or other failures. We describe the characteristics of these patterns of failure and relate them to weak or missing relationships between key artifacts?relationships that should exist even at an early stage of the life cycle. In this paper, we focus on those artifacts that relate to the expression and analysis of business goals. We present early results in the development of acquisition quality attributes (analogous to software quality attributes) and how these attributes relate to acquisition strategies. We conclude with some speculation on what is needed to avoid the failure patterns.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**27**	

The research presented in this report was supported by the Acquisition Research Program of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

Preface & Acknowledgements

Welcome to our Tenth Annual Acquisition Research Symposium! We regret that this year it will be a "paper only" event. The double whammy of sequestration and a continuing resolution, with the attendant restrictions on travel and conferences, created too much uncertainty to properly stage the event. We will miss the dialogue with our acquisition colleagues and the opportunity for all our researchers to present their work. However, we intend to simulate the symposium as best we can, and these *Proceedings* present an opportunity for the papers to be published just as if they had been delivered. In any case, we will have a rich store of papers to draw from for next year's event scheduled for May 14–15, 2014!

Despite these temporary setbacks, our Acquisition Research Program (ARP) here at the Naval Postgraduate School (NPS) continues at a normal pace. Since the ARP's founding in 2003, over 1,200 original research reports have been added to the acquisition body of knowledge. We continue to add to that library, located online at www.acquisitionresearch.net, at a rate of roughly 140 reports per year. This activity has engaged researchers at over 70 universities and other institutions, greatly enhancing the diversity of thought brought to bear on the business activities of the DoD.

We generate this level of activity in three ways. First, we solicit research topics from academia and other institutions through an annual Broad Agency Announcement, sponsored by the USD(AT&L). Second, we issue an annual internal call for proposals to seek NPS faculty research supporting the interests of our program sponsors. Finally, we serve as a "broker" to market specific research topics identified by our sponsors to NPS graduate students. This three-pronged approach provides for a rich and broad diversity of scholarly rigor mixed with a good blend of practitioner experience in the field of acquisition. We are grateful to those of you who have contributed to our research program in the past and encourage your future participation.

Unfortunately, what will be missing this year is the active participation and networking that has been the hallmark of previous symposia. By purposely limiting attendance to 350 people, we encourage just that. This forum remains unique in its effort to bring scholars and practitioners together around acquisition research that is both relevant in application and rigorous in method. It provides the opportunity to interact with many top DoD acquisition officials and acquisition researchers. We encourage dialogue both in the formal panel sessions and in the many opportunities we make available at meals, breaks, and the day-ending socials. Many of our researchers use these occasions to establish new teaming arrangements for future research work. Despite the fact that we will not be gathered together to reap the above-listed benefits, the ARP will endeavor to stimulate this dialogue through various means throughout the year as we interact with our researchers and DoD officials.

Affordability remains a major focus in the DoD acquisition world and will no doubt get even more attention as the sequestration outcomes unfold. It is a central tenet of the DoD's Better Buying Power initiatives, which continue to evolve as the DoD finds which of them work and which do not. This suggests that research with a focus on affordability will be of great interest to the DoD leadership in the year to come. Whether you're a practitioner or scholar, we invite you to participate in that research.

We gratefully acknowledge the ongoing support and leadership of our sponsors, whose foresight and vision have assured the continuing success of the ARP:

- Office of the Under Secretary of Defense (Acquisition, Technology, & Logistics)
- Director, Acquisition Career Management, ASN (RD&A)
- Program Executive Officer, SHIPS
- Commander, Naval Sea Systems Command
- Program Executive Officer, Integrated Warfare Systems
- Army Contracting Command, U.S. Army Materiel Command
- Office of the Assistant Secretary of the Air Force (Acquisition)
- Office of the Assistant Secretary of the Army (Acquisition, Logistics, & Technology)
- Deputy Director, Acquisition Career Management, U.S. Army
- Office of Procurement and Assistance Management Headquarters, Department of Energy
- Director, Defense Security Cooperation Agency
- Deputy Assistant Secretary of the Navy, Research, Development, Test, & Evaluation
- Program Executive Officer, Tactical Aircraft
- Director, Office of Small Business Programs, Department of the Navy
- Director, Office of Acquisition Resources and Analysis (ARA)
- Deputy Assistant Secretary of the Navy, Acquisition & Procurement
- Director of Open Architecture, DASN (RDT&E)
- Program Executive Officer, Littoral Combat Ships

James B. Greene Jr.
Rear Admiral, U.S. Navy (Ret.)

Keith F. Snider, PhD
Associate Professor

Acquisition Management

Naval Ship Maintenance: An Analysis of the Dutch Shipbuilding Industry Using the Knowledge Value Added, Systems Dynamics, and Integrated Risk Management Methodologies

David N. Ford, Thomas J. Housel, and Johnathan C. Mun
Naval Postgraduate School

Time as an Independent Variable: A Tool to Drive Cost Out of and Efficiency Into Major Acquisition Programs

J. David Patterson
National Defense Business Institute, University of Tennessee

The Impact of Globalization on the U.S. Defense Industry

Jacques S. Gansler and William Lucyshyn
University of Maryland

Bottleneck Analysis on the DoD Pre-Milestone B Acquisition Processes

Danielle Worger and Teresa Wu, *Arizona State University*
Eugene Rex Jalao, *Arizona State University and University of the Philippines*
Christopher Auger, Lars Baldus, Brian Yoshimoto, J. Robert Wirthlin, and John Colombi, *The Air Force Institute of Technology*

Software Acquisition Patterns of Failure and How to Recognize Them

Lisa Brownsword, Cecilia Albert, Patrick Place, and David Carney
Carnegie Mellon University

Fewer Mistakes on the First Day: Architectural Strategies and Their Impacts on Acquisition Outcomes

Linda McCabe and Anthony Wicht
Massachusetts Institute of Technology

The Joint Program Dilemma: Analyzing the Pervasive Role That Social Dilemmas Play in Undermining Acquisition Success

Andrew P. Moore, William E. Novak, Julie B. Cohen, Jay D. Marchetti, and Matthew L. Collins
Software Engineering Institute, Carnegie Mellon University

Acquisition Risks in a World of Joint Capabilities: A Study of Interdependency Complexity

Mary Maureen Brown
University of North Carolina Charlotte

Leveraging Structural Characteristics of Interdependent Networks to Model Non-Linear Cascading Risks

Anita Raja, Mohammad Rashedul Hasan, and Shalini Rajanna
University of North Carolina at Charlotte
Ansaf Salleb-Aoussi, *Columbia University, Center for Computational Learning Systems*

Lexical Link Analysis Application: Improving Web Service to Acquisition Visibility Portal

Ying Zhao, Shelley Gallup, and Douglas MacKinnon
Naval Postgraduate School

Capturing Creative Program Management Best Practices

Brandon Keller and J. Robert Wirthlin
Air Force Institute of Technology

The RITE Approach to Agile Acquisition

Timothy Boyce, Iva Sherman, and Nicholas Roussel
Space and Naval Warfare Systems Center Pacific

Challenge-Based Acquisition: Stimulating Innovative Solutions Faster and Cheaper by Asking the Right Questions

Richard Weatherly, Virginia Wydler, Matthew D. Way, Scott Anderson, and Michael Arendt
MITRE Corporation

Defense Acquisition and the Case of the Joint Capabilities Technology Demonstration Office: Ad Hoc Problem Solving as a Mechanism for Adaptive Change

Kathryn Aten and John T. Dillard
Naval Postgraduate School

A Comparative Assessment of the Navy's Future Naval Capabilities (FNC) Process and Joint Staff Capability Gap Assessment Process as Related to Pacific Command's (PACOM) Integrated Priority List Submission

Jaime Frittman, Sibel McGee, and John Yuhas, *Analytic Services, Inc.*
Ansaf Salleb-Aoussi, *Columbia University*

Enabling Design for Affordability: An Epoch-Era Analysis Approach

Michael A. Schaffner, Marcus Wu Shihong, Adam M. Ross, and Donna H. Rhodes
Massachusetts Institute of Technology

Measuring Dynamic Knowledge and Performance at the Tactical Edges of Organizations: Assessing Acquisition Workforce Quality

Mark E. Nissen
Naval Postgraduate School

Outcome-Focused Market Intelligence: Extracting Better Value and Effectiveness From Strategic Sourcing

Timothy G. Hawkins, *Naval Postgraduate School*
Michael E. Knipper, *771 Enterprise Sourcing Squadron USAF*
Timothy S. Reed, *Beyond Optimal Strategic Solutions*

Software Acquisition Patterns of Failure and How to Recognize Them[1]

Lisa Brownsword—Brownsword is a senior member of the Acquisition Support Program at the Software Engineering Institute (SEI). She manages and participates in customer engagements for major programs within the DoD and federal agencies, providing pragmatic expertise in software engineering, systems of systems (SoS), commercial-off-the-shelf (COTS)-based systems (CBS), architecture and product lines, and iterative development. She is managing a research team to identify patterns of misalignment of acquisition strategies, architecture, and program business and mission goals that lead to program problems. [llb@sei.cmu.edu]

Cecilia Albert—Albert is a senior member of the technical staff in the Acquisition Support Program at the SEI, where she has managed strategic software improvement across Army programs and co-developed a process for developing COTS-based systems. Albert has more than 35 years of experience developing and acquiring software-reliant systems and holds a master's degree in computer science from Stanford University and a bachelor's degree from Sweet Briar College. [cca@sei.cmu.edu]

Patrick Place—Place is a senior member of the technical staff in the Acquisition Support Program at the SEI. Recent projects include developing practices for engineering in an SoS context; the Service Migration and Reuse Technique (SMART); and the acquisition implications of adopting a service-oriented architecture (SOA) development strategy. He has participated in a number of independent technical assessments related to the adoption of SOA practices. Place has been an adjunct lecturer at Carnegie Mellon University, South Bank University, and Imperial College, teaching various courses on the use of formal specification techniques. [prp@sei.cmu.edu]

David Carney—Carney is a retired member of the SEI, now working as a consultant on select Acquisition Support Program assignments. He has 30 years of experience as a software engineer working at Intermetrics Inc., the Institute for Defense Analyses, and the SEI. His fields of particular interest have included CASE tool environments, SoS integration, issues relating to the use of COTS products, and service-oriented architectures. [djc@sei.cmu.edu]

Abstract

In systems today, software provides substantial portions of capability and performance. In many Department of Defense (DoD) acquisitions, however, software too often is a minor consideration when the early and most constraining decisions about program cost, schedule, and behavior are made (i.e., prior to Milestone A). These decisions manifest themselves in the acquisition strategy, the system and software architecture, and ultimately in the deployed

system. Based on our experience with large programs, we have identified seven patterns of failure that lead to misalignment between the software architecture and the acquisition strategy, leading to program restarts, cancellations, or other failures.

We describe the characteristics of these patterns of failure and relate them to weak or missing relationships between key artifacts—relationships that should exist even at an early stage of the life cycle. In this paper, we focus on those artifacts that relate to the expression and analysis of business goals. We present early results in the development of acquisition quality attributes (analogous to software quality attributes) and how these attributes relate to acquisition strategies. We conclude with some speculation on what is needed to avoid the failure patterns.

Introduction

In systems today, software provides substantial portions of capability and performance. One consequence for DoD acquisition is that software issues are major factors in cost and schedule overruns. However, software is often a minor consideration when the early, most constraining program decisions are made. This has negative consequences, most often in terms of *misalignments* between the software architecture[2] and system acquisition strategies.[3] Our analysis of troubled programs shows that these misalignments lead to program restarts, cancellations, and failure to meet important mission and business goals.[4] Our research focuses on enabling organizations to reduce their program failures by harmonizing their acquisition strategy with their system and software architectures.

One major source of misalignments is the diverse sets of stakeholders of complex programs; it is inevitable that some (perhaps many) of their diverse goals and priorities are in conflict. Operational users, combat commanders, funding authorities, and acquisition team members may think they share the same priorities, but when interviewed, their answers often vary widely in terms of the goals and features they see as the most important. In many cases, the solutions that are then created are based on goals of one set of stakeholders—goals that can conflict with those of other stakeholders without explicit consideration of the tradeoffs being made. Ultimately, such conflicts in goals eventuate in the misalignments described previously.

Observed Patterns of Failure

Characteristics of the Patterns We Observed

The major source of our data for this research was interviews with participants in major acquisition programs, most of which encountered some sort of difficulty, ranging from partial to total failure. These data are also supported by decades of experience on the part of all of the authors in studying, assessing, and participating in actual programs, many of which evidenced similar failing behaviors. Virtually all of the conclusions derived from our

---

[2] *Software architecture* is defined by Bass, Clements, and Kazman (2012) as “the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.”

[3] *Acquisition strategy* (2011) is defined by the Defense Acquisition University as “a business and technical management approach designed to achieve program objectives within the resource constraints imposed. It is the framework for planning, directing, contracting for, and managing a program. It provides a master schedule for research, development, test, production, fielding, modification, postproduction management, and other activities essential for program success.”

[4] A mission goal is an expression of an objective that affects a user, focused on what the solution or product should do or how it should behave. A business goal is an expression of an organizational (e.g., Navy) objective, focused on goals relative to the business model for acquiring or developing the solution or product.

interviews were strongly supported by our aggregate experiences as active professionals, in both government and non-government roles in the domain of DoD acquisition. Analysis of these data has led us to observe several recurring patterns. Because of their negative consequences, and following common usage throughout the software engineering community as exemplified by Brown, Malveau, McCormick, and Mowbray (1998), we characterize these as *anti-patterns*.

Buschmann, Meunier, Rohnert, Sommerlad, and Stal (1996) provided a form for describing patterns, calling out the need to give patterns a name and to define the context (environment), the problem, and the solution. We modify this approach for our anti-pattern descriptions. First, we include the notion of consequence, as defined by Gamma, Helm, Johnson, and Vlissides (1994). Second, in lieu of calling the observed activity "solution," we have titled that activity "observed response" (i.e., to the problem). Thus, each of our anti-pattern descriptions has the following elements:

- name (readily identifies some key aspect of the problem)
- context (situations where the pattern occurs)
- problem (recurring issues and forces that characterize the problem)
- observed response (the manner in which people attempt, consciously or otherwise, to solve the problem)
- consequences (results of applying the observed response to the problem in the given context)

Overview of Findings

In each of the programs studied, we observed several instances of activities and behaviors that qualify as anti-patterns. In some cases, the anti-patterns were of sufficient magnitude that they had severe negative effects on program success. None of the anti-patterns were observed in all of the programs we studied, and none were seen in only one program. We, therefore, believe that they were sufficiently pervasive that they were true patterns of behavior.

The set of anti-patterns we observed in these programs consists of the following:

1. Undocumented Business Goals
2. Unresolved Conflicting Goals
3. Failure to Adapt
4. Turbulent Acquisition Environment
5. Poor Consideration of Software
6. Inappropriate Acquisition Strategies
7. Overlooking Quality Attributes

These anti-patterns, like most factors that negatively affect acquisition programs, eventually result in a small number of familiar and unhappy consequences: schedule delays, cost overruns, delivery of less than was promised, or outright program cancellation. In that sense, the *ultimate* consequences of these anti-patterns in the programs we studied have an expected similarity.

However, the *immediate* consequences of different anti-patterns differ in many ways. For example, the immediate effects of leaving key business goals unstated is different from

those that result from a turbulent acquisition environment. Since, in the methods we hope to develop, we intend to focus on some of the immediate and visible symptoms of anti-patterns as a way to minimize or eliminate their negative influence, these immediate consequences are important. Therefore, in the descriptions that follow, we indicate a number of these immediate consequences, although we also mention, wherever possible, the longer term consequences that we observed.

Each of these is discussed in the subsections that follow.

Undocumented Business Goals

Context. This anti-pattern stems from a lack of precise, well-defined, and well-documented business goals for a DoD acquisition, goals that would correspond to the precise, well-defined mission goals usually created for a program. The DoD's business goals are the major driver for an acquisition strategy, which should make some provision for software. But the actual role that the detailed business goals play in the software, particularly its architecture, is often minimal.

Problem. Although business goals obviously influence a program's acquisition strategy, they can also have a strong influence on system and software architecture. This additional influence is seldom recognized, even when it is vital that it should be. Examples of such influence include the following:

- "avoid vendor lock" or "maximize competition" has potentially significant importance when defining software architecture;
- a mandate to employ reuse as much as possible may have a strong negative impact on the software architecture if the software to be reused is itself poorly architected; and
- the goal of a *software* "open architecture" may have a significant impact on the underlying *system* architecture.[5]

Two factors cause this problem. First, at the high level, many business goals are generally expressed only as very broad mandates, and others are not explicitly expressed at all. Second, at the detailed level, there is no useful process for capturing and prioritizing business goals in a program-specific way that is comparable to the Joint Capabilities Integration and Development System (JCIDS) process that supports definition and prioritization of mission goals and their associated requirements.

This anti-pattern is particularly problematic in programs that are building a system that must integrate with other systems (which may themselves be in varying stages of development). While the high-level goal of an integrated system may be explicit, the detailed goals (together with an understanding of needed resources) for the system of systems (SoS) are often left unspecified.

Observed Response to the Problem. The general response to this anti-pattern is that the architect has no other choice, and hence the mission requirements defined by the operational side drive the architecture, which then reflects *only* those mission requirements. Yet, were the detailed business goals available, some, perhaps many, of those business goals might be critical enough to overshadow some of the mission requirements.

[5] Using Maier (2006), we characterize typical system architectures as compositional (is-a-part-of relationships) and software architectures as more typically layered hierarchies (is-used-by relationships).

An example can be seen in an instance where there might be an implicit but unspecified business goal that favored a highly distributed contractor/subcontractor profile. Lacking awareness of that goal (because it is unstated), a software architect might reasonably design a monolithic architecture to satisfy a mission goal for performance.

Consequences. This anti-pattern was observed in three of the six programs under study. In the program in which it was most visible, a new system with significant new capabilities was needed; a business goal was to replace several systems that were "end-of-life." The acquisition strategy specified a slow, deliberate pace to ensure that the new capability was defined correctly. But ignored in this strategy was the urgent need by users to replace these failing systems as quickly as possible. When the operators and maintainers of the legacy systems became aware of the intended acquisition strategy, they forced a major change in focus for the program. The consequence was a significant delay for the program.

In the other programs where this anti-pattern was observed, the effect was less pronounced. Systems were delivered, although with less functionality than was expected. In all cases, follow-on programs have been started to create the functionality that was originally promised by these programs.

In sum, the overall effect of this anti-pattern is that important guiding documents—in particular, the architecture and acquisition strategy, or both—fail to reflect the *joint* influence of both business and mission goals. Inevitably, the lack of documentation of missing business-related goals (and their associated quality attributes for the architecture) will result in a system that fails to meet the expectations of at least some of the key stakeholders, because the joint expectations of *all* of the stakeholders have never been adequately reasoned about.

Unresolved Conflicting Goals

Context. This anti-pattern is often a direct consequence of the previous one, the distinction being that the first anti-pattern refers to the *absence* of well-documented business goals, while this one refers to the *lack of an analysis and de-confliction* of the known goals.

Problem. The variety and scope of mission and business goals can be very large, and for a program of any consequence, there will likely be conflicts among some of these diverse goals and priorities. One factor that compounds the problem is that the business goals and the mission goals are often developed by people from different communities—people with very different concerns.

But to reason about these conflicts requires that all of these goals be considered jointly, so that their mutual influence can be understood and misalignments negotiated. Reasoning is obviously impossible if, as in the previous anti-pattern, the business goals are not well documented. But even if there is a set of well-documented business goals, no processes or criteria exist by which tradeoffs between important business and mission goals can be made. It is often not even clear who should arbitrate such goal conflicts.

A frequently observed example of this anti-pattern is reflected in the conflict between the goal of introducing a new or updated system and the additional goal of avoiding impacting how current end users perform their tasks. At best, this is a conflict of expectations that is not fully understood until the system is deployed, potentially presenting a barrier to deployment.

Finally, one specific problem that is often observed, and one that mixes both business and mission elements, is that a program shares dependencies with other systems in a larger SoS context. Too often, each of these systems is considered in isolation, and the mission and business effects that each program should have on the others is ignored. When

these effects surface, joint consideration is often carried out too late in program execution to be effective or to succeed.

Observed Response to the Problem. Program personnel tend to separate into business (e.g., acquisition strategy) and mission (e.g., system and software architecture) communities, each of which tends to work in isolation. Given this separation, these personnel tend to produce artifacts that reflect the goals and priorities known to them; these in turn may be misaligned in that they reflect unresolved conflicts between business and mission goals such as those described previously.

Consequences. This anti-pattern was observed in five of the six programs under study. In one program, various business goals concerning reuse, using multiple contractors, reducing integration costs, and such mission goals as greatly increased performance had produced numerous unresolved conflicts. When the gravity of the conflicts was belatedly understood (by an independent "tiger team"), both the initial acquisition strategy and the initial architecture were abandoned, and a major reconsideration of both—in which they would be reconciled and aligned—was begun. While this brought about a significant delay, it avoided the far worse result of a system that failed both its business and mission stakeholders.

In general, it is likely that in a program of any consequence, conflicts between business and mission goals will exist. But if the conflicts are not reconciled before the acquisition strategy and the architecture (both system and software) are defined, the negative effects that these conflicts will have will be large. Unless *joint* consideration of mission and business goals is carried out early in program execution, conflicts between goals will soon become difficult, or even impossible, to reconcile. And, ultimately, stakeholders who expected their mission or business goals to be reflected in the acquisition strategy and then satisfied by the software architecture are unhappily surprised when the system cannot support their mission or business objectives.

Failure to Adapt

Context. This anti-pattern often occurs when program duration is very long. The reasons for length can be inherent, such as when a system is unprecedented and requires considerable time to solve massive engineering problems (for instance, creation of the Joint Strike Fighter). Or the reasons for highly extended program duration can be circumstantial, such as a protracted, complex protest to a contract award. This anti-pattern can also occur when a program evolves from providing limited capabilities to providing a much wider range of capabilities.

Problem. In most programs, both the acquisition strategy and the architecture are optimized to meet the goals and priorities that exist at the start of the program. However, goals and priorities naturally evolve over time: Examples of such change could include the need to combat new and unexpected threats, or a desire to modernize a capability using new technology. The essential problem lies in the fact that the architecture and acquisition strategy that are initially defined may not be flexible enough to respond to these changes without a good amount of revision and redefinition.

Further, and compounding the problem, there are no widely applicable processes for rapidly revising acquisition strategy for changed business goals, nor are there widely used processes to accommodate changes to architecture as a result of such changed goals.

Observed Response to the Problem. When such changes as those described previously occur, program personnel are often unsure about whether the architecture, acquisition strategy, or both can accommodate the needed changes and, even if they can,

whether the changes can be accomplished, given the time and effort that will be required (e.g., to get all necessary approvals for a revised acquisition strategy). Hence, programs tend to continue executing as though there has been minimal change to the initial goals and priorities; there is little impetus to revise the acquisition strategy or make anything more than minimal alterations to the architecture. In effect, the program is operating with either an implicit change in acquisition strategy or a mismatch between the architecture defined initially and the changed mission goals, or both.

Consequences. In one of the programs we studied, this anti-pattern had a considerable negative effect. The program initially had a very successful architecture and acquisition strategy; the program was so successful that its scope grew from delivering capability for one system to delivering capability to multiple systems of a similar type. The architecture and acquisition strategy survived this change in scope initially, but as the separate systems matured, and as need arose for them to interoperate, unexpected demands were placed on the architecture. An additional factor was that the stakeholders became increasingly interested in system reliability—a new quality attribute for this program. At this point, both the architecture and the acquisition strategy failed. This program has entered a strategic pause while the architecture and the acquisition strategy are reconsidered in light of both the new requirement (reliability) and the increased complexity of what is now a SoS.

In general, this anti-pattern reflects a natural tendency to stay the course, even when circumstances change and external conditions evolve.

Turbulent Acquisition Environment

Context. This anti-pattern is closely related to anti-pattern #3 (Failure to Adapt). But in this case, the cause is not extended program evolution but rather severe instability in multiple program elements. This instability is manifest by changes in goals, strategy, or architecture that are so frequent and contradictory, they require adaptation that, even under the best circumstances, the program is unable to accommodate. These changes can be political, strategic, technological, or fiscal.

Problem. Several causes can bring about program turbulence. Budgets can undergo major revisions, and major portions of a program's funding can be withdrawn. Mission circumstances can be suddenly changed, or radically new technologies can be disseminated rapidly. Programs, particularly if they are perceived to be in severe difficulty, can face significant revisions of goals and purpose. Joint programs often undergo periodic management shifts when different services assume primary responsibility.

In cases where one or more of the previously mentioned conditions are present, a program can find itself faced with the demand for change that is impossible to accommodate. The magnitude of the requested change is often unrealistic, impractical, or impossible, given time and resource realities.

As the program personnel attempt to adapt to the changes, the original architecture and acquisition strategy may now be highly unsuited to the changed conditions that have been levied on the program. The program falls into a mode of "architecture of the day" or "acquisition strategy of the day." Equally problematic and an important part of the problem is that the program is usually still contractually held to some part of the original acquisition strategy.

Observed Response to the Problem. The frequent and significant changes in mission or business goals overpower the ability of the acquisition strategy or architecture (or both) to accommodate them. Thus, the original acquisition strategy is implicitly abandoned

but without having a well-defined new strategy created. Or the architecture is stretched to the breaking point and loses all relation to the original acquisition strategy.

Yet many programs attempt to continue executing with, at most, minimal explicit revision of the acquisition strategy and/or architecture. The necessary task to revise the acquisition strategy to fully account for the changed goals is seldom performed, nor is the work needed to revise the original architecture carried out as carefully as is required.

Consequences. This anti-pattern was observed in three of the six programs under study. In one of them, significant changes to mission, architecture, hardware, and program direction each occurred, some repeatedly. For instance, the program began with a strong research basis but very quickly was given mandates to field equipment as quickly as possible. Different quality attributes were given different priorities as the architecture evolved through several iterations of development. Different contractors were given conflicting priorities throughout program execution. The result was that the program fielded a small fraction of what was originally planned, after which the program was cancelled.

There is little doubt that the environment of DoD acquisition always has some instability; that is the natural condition of government programs. But in the final analysis, when the environment is truly turbulent (i.e., when this anti-pattern is strongly present), the best result is likely to be systems that are poorly fitted to the purposes for which they are to be used. In the worst case, they may even be unfit for use at all.

Poor Consideration of Software

Context. This anti-pattern occurs when critical decisions are made, especially early in a program, that have strong negative implications on the system's software. There is a historical basis for this behavior: For decades, the DoD acquired systems that were primarily hardware. But while the role and importance of software has grown significantly in recent years, the traditions and habits of acquisition still reflect the earlier, hardware-centric posture.

Problem. Very often, software is not deemed a critical factor in decisions made at the earliest stages of a program. These decisions generally are made with little or no understanding of how software must be accounted for in the acquisition strategy or the architecture (or both).

One symptom: Contracts are organized based primarily on the system architecture and fail to take into consideration the very sizable role of software. Assumptions are made about the expected integration of software entities that are created separately; such assumptions are often made with no understanding of the difficulties that arise when complex independent software systems must interoperate or be integrated. This leads to system architectures and acquisition strategies that over-constrain the yet-to-be-defined software architecture, thus adding significant complexity to software development and integration. Even in a software-only system, the real difficulties that the software can pose (e.g., integration of many heterogeneous components) are largely ignored.

One other such symptom is that quality attributes that are important to the system engineering community may be quite different from those that are significant to the software community. Even if the two communities speak of a single quality attribute (e.g., reliability), they refer to different things. Thus, early decisions about quality attributes typically are decisions about system, not software, quality attributes.

Observed Response to the Problem. As a result, the acquisition strategy either is created with a strong focus on system architecture or, in software-only systems, fails to address the software architecture satisfactorily. In the former case, this inevitably produces

a large gap between the system and software architectures; in the latter case, the planned software acquisition strategy is unrealistic and impractical.

A common “solution” is that the software architect tries to play “catch up” and fit the software to the system architecture. Another “solution” is to ignore the eventual complexities of integration and expect that they can be resolved later. In these cases, the result is often that the system constraints force software choices that are suboptimal for the whole system.

Consequence. This anti-pattern was observed in three programs. In one program, two critical early decisions about software were made with very little understanding of software’s inherent complexity. The system was large and complex, with three major software components. An early decision concerning the integration of those three components downplayed all of the details of that integration: who would do it, what resources it would need, and how difficult it would be. No attempt was made to base this decision on expert software advice, nor on data readily available from comparable programs about the difficulty of such a task. Another decision related to the assumption that the system could later be made to interoperate with another complex software system. But no rigorous assessment was made of the difficulty of accomplishing that interoperation nor of the resources that this integration would require. In both cases, the assumptions proved false, and the program was cancelled.

In general, in the presence of this anti-pattern, there will be major software requirements that cannot be satisfied. In a system with both hardware and software elements, this is often a direct result of a software architecture that had to be made to fit poorly into a system architecture that was already defined. Further, the problems and delays that result from the gap between the system and the software architectures are typically blamed on the software components alone. In the worst cases, these suboptimal decisions are reflected in system-level schedule delays and cost overruns.

Inappropriate Acquisition Strategies

Context. The time factor figures prominently in DoD acquisitions. Starting from the earliest moments of a program (i.e., the awareness of a need for a new or updated system), one urgent, yet often unstated, imperative is to move quickly to avoid any eventualities that might delay, or even prevent, a program from achieving its desired milestones. This imperative is often exacerbated by the need to spend an amount of money that was established as many as two years previously, at a point where little was actually known about the realities that the program would face. It is further exacerbated by the lengthy review process before a new contract can be awarded. These realities of the time factor have led to acquisition strategies that are often poorly suited to an individual program’s needs.

Problem. There are multiple causes of this anti-pattern. Program offices might

- wish to avoid protests
- get quick approval for a program (e.g., before anticipated budget cuts)
- lack sufficient acquisition expertise to develop an acquisition strategy that will quickly gain approval
- have a particular acquisition strategy imposed by a higher authority

In another scenario, some key business goals (e.g., split an acquisition that is conceptually a single system into multiple acquisitions to avoid a “big bang”) are in direct conflict with the technology to be used, the system to be built, or both.

Note that these causes can be either external (protests, budget cuts) or internal (inexperience of a program management office to develop and defend a solid acquisition strategy).

Observed Response to the Problem. Whatever the particular cause, the result is that the primary goal of the program office and the source selection team is to get through a competition and issue a contract as quickly as possible. And even though the chosen acquisition strategy might have a poor fit, or even a misfit, with the business and/or mission goals for the system, the inappropriate acquisition strategy is put in place. The program begins execution, deferring or ignoring the parts of the strategy that have a poor fit with the real needs of the system to be built and, too often, the wrong contract relationship with the software developer(s).

Consequences. This anti-pattern was observed in five of the six programs under study. In one program, the fear of a "big bang" approach led to splitting the intended large system into two separate acquisitions, with the assumption that the two systems could later be integrated into a large SoS. But the second program suffered a very long and complex protest and did not get underway until several years after the first had begun. By that time, the first had completed most of its requirements and was nearing initial operational deployment. But there had been no input from the stakeholders of the second system about the interfaces that would be needed to make eventual integration of the two systems feasible. Thus, the plan for integrating the two systems was abandoned, and the second program was eventually cancelled. The first program fielded a system that provided only a small portion of the expected functionality.

In general, in the presence of this anti-pattern, one of two immediate consequences will emerge. Either the acquisition strategy is ignored as the program unfolds or the program is forced to bend the needs of the system to the inappropriate strategy. In the latter case, the system can reach a point where it can no longer meet its mission goals, business goals, or both.

Overlooking Quality Attributes

Context. In the earliest stages of a program's life, there may be no formal program office and only minimal accompanying funding to perform necessary work. Further, there is significant pressure to rapidly produce the acquisition strategy and initial architecture in order for the program to be funded. But there is no requirement to use quality attributes to define that architecture, and little incentive to do so. Further, in many cases, the detailed business goals are unwritten (see anti-pattern #1) or the importance of the software is ignored (see anti-pattern #5), and hence, there is little opportunity to expose the quality attributes that the system is expected to manifest.

Problem. The program overlooks the software quality attributes that should support the goals, whether mission or business. Instead, programs rely on key performance parameters (KPPs), which are not broken down in sufficient detail that architects can reason about the necessary alignment among the software architecture, system architecture, acquisition strategy, and aggregate set of goals for the system.

Observed Response. In order to meet the reporting needs and get to a "go" or "no go" decision for a system, programs put their engineering resources into eliciting and capturing the functional capabilities and requirements and provide only minimal attention to quality attributes by focusing on a limited set (e.g., performance, availability). Worse, the notions of those quality attributes are more often those understood by system engineers rather than software engineers.

As a result, architectural decisions that should be based on extensive consideration of all quality attributes—software as well as systems—are made by "gut feel," or by adopting an architecture from a similar or idealized system, rather than by reasoning that placed real importance on the specific software quality attributes for the system at hand.

Consequences. This anti-pattern was observed in four of the programs under study. In one joint program, the system was to be integrated into operations in each of the military Services. However, the concept of operations for each of the Services was different (i.e., where the system would be hosted, security needs, what other systems would be integrated), and these differences, all of which implied different quality attributes, were not recognized early in the program. Neither the acquisition strategy nor the architecture accommodated these differences. The quality attribute descriptions that were constructed reflected the needs of only one of the Services and focused only on technical issues; they explicitly ignored that same Service's business goals. It eventually became apparent that the program office and the end users had very different approaches to meeting even the single Service's stated needs.

In general, the primary consequence of this anti-pattern is that the resultant system architecture (and the likely inefficient software architecture) will satisfy only some of the goals; others will be, at best, partially satisfied and often unsatisfied. In the longer term, since sufficient knowledge of the quality attributes is lacking, the program will not have the strong analytic base needed to fully understand the impacts of different modes of evolution that might be needed over the system's life cycle.

Countering the Patterns

The anti-patterns described in the preceding section provide evidence of undesirable behaviors that are repeated across multiple programs. We believe that these undesirable behaviors are not intentional but indicate flaws in the existing approach to the acquisition of software-reliant systems. We also believe that at least part of the solution to this problem lies in analyzing how these anti-patterns operate at both the micro level and the macro level. In the previous section, we examined the individual consequences of each anti-pattern. In this section, we consider how, in combination, they jointly can affect the major entities that participate in the acquisition process.

Necessary Entities and Artifacts

The anti-patterns we observed relate to a small number of distinct entities, each of which has major importance for a program and the system it is building. For instance, one anti-pattern focused on how programs ignored the impact of quality attributes. There is ample evidence and experience, such as that discussed by Gagliardi, Wood, Morrow, and Klein (2010), showing that the main drivers for software architecture should be the quality attributes that the system must exhibit. These quality attributes are derived from another key entity, the mission needs expressed by stakeholders.

Another anti-pattern dealt with business goals that were either only expressed generally or only implicitly understood. The business goals for a program are an additional key entity. But these are likely the goals of a very different set of stakeholders. Although not commonly understood, the business goals, like the mission goals, will have quality attributes that should be the main drivers for the acquisition strategy; we assert that these other quality attributes are as important as those derived from the mission goals. We will refer to these other quality attributes as *acquisition quality attributes*.

Given these two sets of goals, which in turn are the principal drivers for two very critical entities (i.e., the acquisition strategy and the software and system architectures), the

potential for conflicts between those goals is large, as is shown in the anti-pattern of conflicting goals.

Finally, the notion of "the stakeholders" embodies a complex and diverse collection of individuals and organizations; they are the sources for the goals and the recipients of the benefits of the system to be created.

We posit several key entities of interest:

- mission goals
- the (mission) quality attributes implicit in those goals
- business goals
- the (acquisition) quality attributes implicit in those goals
- the acquisition strategy
- the software and system architectures, which are closely related, but separate
- the different sets of stakeholders who have expressed needs that are captured by the mission and business goals

While these entities represent a diverse set of things, including humans (stakeholders) and intangibles (goals), we posit that all of these entities must, at least in some manner, be manifest in physical artifacts. Some such artifacts are immediately obvious: the mission goals will ultimately be reflected in a requirements specification; an acquisition strategy document is mandatory for any program. Other artifacts are less well defined and may not even be present in a given program. We assert that they are just as necessary.

The stakeholders for a given acquisition, for instance, cannot be a vague collection of unknown persons but must be defined with at least minimal specificity: "the HR personnel who do data entry for the Air Force Logistics Command," "the Assistant Secretary of the Navy for XYZ," and so forth. The definition of the stakeholders may not have a formal document type associated with it, but it must be physical: There must be some way to determine who precisely has one or another goal, if only to assist in determining priorities and negotiating conflicts. Similarly, the business goals themselves may first be only general expressions found in a statement of need. But eventually those must find some form of detailed notation in a physical document that is a real analog to a requirements specification. Thus, we assert that each of the entities listed previously has an associated artifact, which we can inspect, reason about, and compare with other artifacts of the acquisition process.

Necessary Relationships

These entities are related to each other by means of several different relationships. For each, we use the formula "Entity X <relationship> Entity Y." The following are the pertinent relationships identified in our analysis:

- <have>
- <are embodied by>
- <are consistent with>
- <drive>

- <constrains>
- <informs>

We show these entities and relationships in Figure 1. Some relationships are unidirectional, and the arrow indicates which entity is the actor (e.g., for "X <drive> Y," the direction of the arrow is from X to Y). Some of these relationships are reflexive (e.g., "X <informs> Y" *and* "Y <informs> X"). In these cases, the arrow is two-headed.

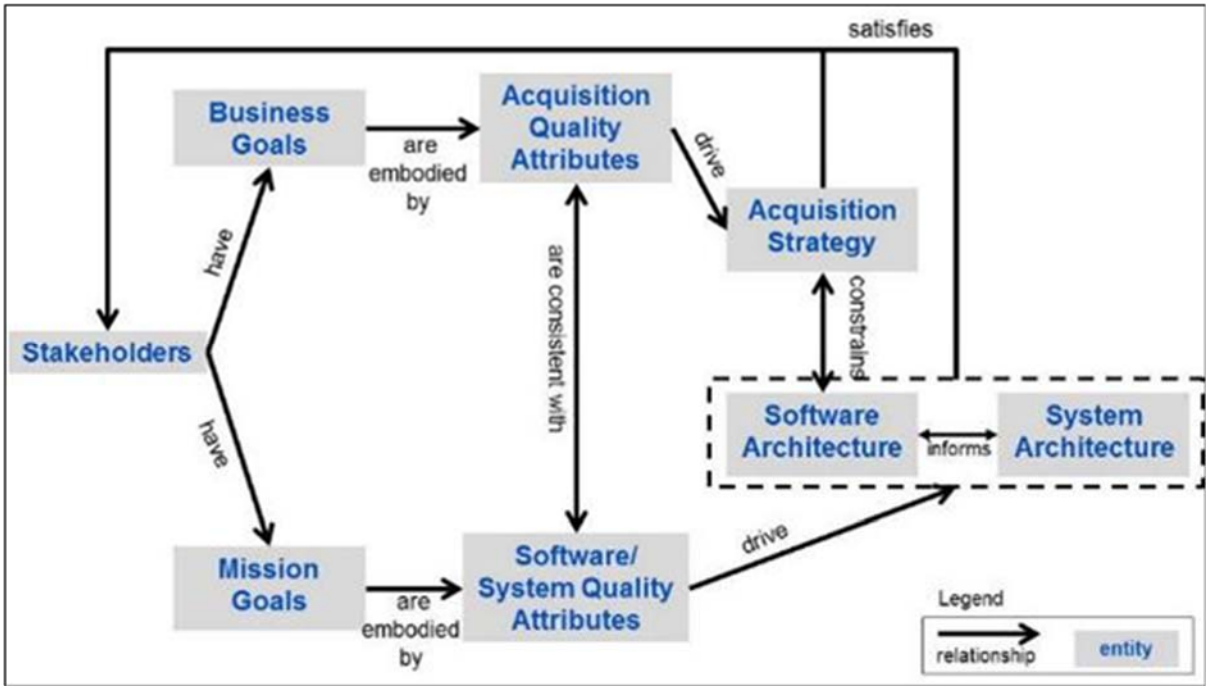

Figure 1. Key Entities and Relationships Identified During Our Analysis

When we consider these entities and relationships in light of the anti-patterns, we see the following:

The <have> relationships concern anti-patterns #1 and #4. While stakeholders have business goals, these goals are often not expressed; the problem is exacerbated by the lack of a process for recording business goals. If these goals can be captured so that the collection of business goals, stemming from all stakeholders, exists in a coherent document, then anti-pattern #1 cannot occur; the business goals will have been documented. If a program office captures and records these goals, then the office can reason about changes in the acquisition environment. It can determine whether the inevitable changes can be accommodated within the program's current scope or whether a reset of the acquisition strategy or the architectures or both will be required. If a program office can perform such reasoning then, although turbulence in the acquisition environment cannot be prevented, the program office can have an appropriate response to the turbulence and prevent the occurrence of anti-pattern #4.

The <are embodied by> relationships concern anti-pattern #7. If the business and mission goals are analyzed and re-expressed in terms of acquisition quality attributes and software/system quality attributes, respectively, then anti-pattern #7 cannot occur since the quality attributes will have been carefully analyzed as part of program creation.

The <are consistent with> relationship concerns anti-pattern #2. Assuming that all of the relevant quality attributes (software, system, and acquisition) have been clearly expressed, it will be possible to reason about all quality attributes, comparing and performing tradeoffs between the types of attributes. If the quality attributes are consistent with each other, then conflicts among the goals will have been resolved and anti-pattern #2 will not occur.

The <drive> relationships concern anti-patterns #5 and #6. Years of research and practical usage of quality attributes have demonstrated that they should be the primary influences on both software and system architectures (Bass, Clements, & Kazman, 2012). In the situations where the quality attributes are used to create the architectures, then we can be certain that those qualities important to the program have been considered and the resultant architecture is consistent with the quality attributes. In such a case, anti-pattern #5 cannot occur. Similarly, if the acqusition strategy is derived from the quality attributes that have been developed from the business goals, then it is likely that the strategy will indeed reflect all of the stakeholder expectations and cannot be considered to be inappropriate to the institutional goals of the organizations involved, thus preventing the occurrence of anti-pattern #6.

The <constrains> relationship concerns anti-pattern #3. Even when the acquisition strategy and the software architecture are specifically aligned, this alignment must be maintained through the life of the system and/or the life of the program office. As the system matures, new goals emerge that must be accommodated in the acquisition strategy or the architecture or both. Ensuring that the architectures continue to constrain the acquisition strategy and the acquisition strategy continues to constrain the architectures will increase the likelihood that the program is feasible. In such a way, anti-pattern #3 can be prevented from occurring.

The <informs> relationship does not specifically concern an observed anti-pattern. Nonetheless, considerable research has shown the importance of mutual influence between software and system architecture. Maier (1998) succinctly described the different perspectives and the impact of not explicitly balancing the engineering and managing between these two architectures. We therefore include it here for completeness.

Figure 2 summarizes the anti-patterns as they affect the entities and relationships.

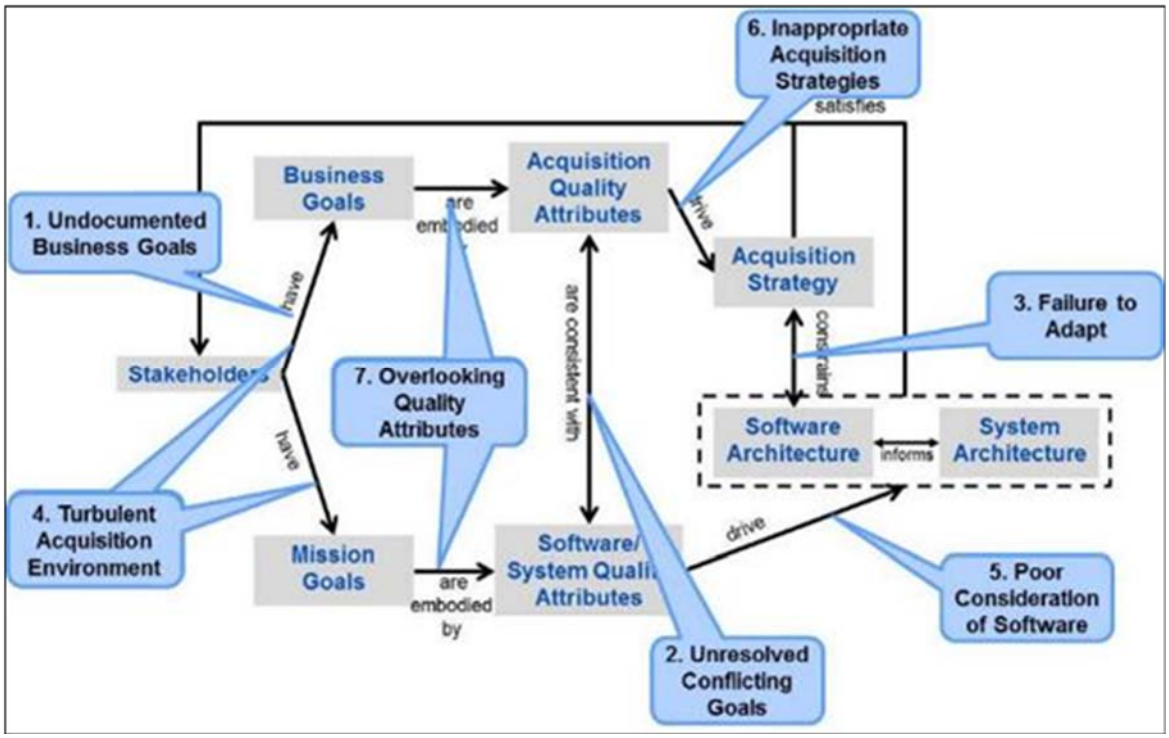

Figure 2. Anti-Patterns That Affect Specific Relationships and Entities

Acquisition Quality Attributes

A key aspect in our work is the notion of *acquisition* quality attributes. This represents a new concept and is the focus of our current investigations. We have introduced them because we are motivated by the considerable body of work on *software* quality attributes and expect that the acquisition quality attributes are as important to acquisition strategy as software quality attributes are to the software architecture. Because we see such a close analogy between these two types of quality attributes, we believe that research into these attributes can be modeled on the research into software quality attributes.

Role of Acquisition Quality Attributes

Acquisition quality attributes are derived from the business goals and exert their strongest influence on the acquisition strategy. They relate the business goals to the acquisition strategy in the same way that the software quality attributes relate the mission goals to the software architecture. To formulate acquisition quality attributes for a program implies the following step of explicitly developing an acquisition strategy that accommodates them: They must be designed into the strategy to minimize the risk of failure. Note that incorporating these qualities into the acquisition strategy is no guarantee that the program will be successful, but it lays the foundation upon which success can be built.

As with software quality attributes, acquisition quality attributes will be dependent on the specific circumstances of the program in question. For example, a program may have little concern for long-term sustainability if there is little expectation that the deployed system will be sustained over the long term; hence, the acquisition strategy should be concerned about many quality attributes but *not* about sustainability. (This can occur in situations where there is an imperative need to get something deployed as rapidly as possible.). Or a program may be developed as a research prototype with no concern about deployability or sustainability if the major goal is to prove a concept, not field a capability. The key point is

that these attributes must be understood in the context of each particular program so that a suitable acquisition strategy can be developed.

As with software quality attributes, we expect that some acquisition quality attributes will be universal. Thus, every program is started with some sense of producing a good result, and we would expect every program to have some sense of "achievability," although precisely what that means would differ from program to program. Similarly, programs with the goals for a long lifetime will exhibit sustainability and evolvability; if not, the programs are unlikely to enjoy their long life spans without major, costly adjustments over their lifetimes.

Giving Meaning to Acquisition Quality Attributes

As stated previously, our view of acquisition quality attributes is that they are an analog to software quality attributes, although the qualities they relate to may differ. And like software quality attributes, they have little inherent meaning that is not ambiguous and must be given precise meaning by some other mechanism.

One approach would be to try to develop precise definitions of each acquisition quality attribute. Our experience, however, is that this would lead to the same difficulties found with software or system quality attributes, where many different and inconsistent quality models exist and the definitions are either too general or too contradictory to be of use. (For instance, one could conjecture many definitions of *sustainability*, but which definition was appropriate to a specific program and system would still need to be resolved.)

Instead, we follow the approach used in software architecture (and later, system architecture) of developing scenarios, which have the purpose of defining precisely the meaning of the attributes in the context of a specific acquisition. Such an approach eliminates needless discussion about the "correct" definition of any individual attribute and, more importantly, gives meanings based on the context of a particular acquisition.

The software architecture work at the SEI was of great benefit in representing quality attribute scenarios in six parts: source, stimulus, artifact, environment, response, and response measure (Barbacci et al., 2003). Following this approach, we characterize the parts of an *acquisition* quality attribute in the same manner:

- *Source*: The person, group, or organization that generates the stimulus.
- *Stimulus*: A condition or event that needs to be considered when it occurs during the acquisition.
- *Environment*: The conditions under which the stimulus occurs.
- *Artifact*: The artifact that is stimulated.
- *Response*: The activity undertaken after the arrival of the stimulus.
- *Response measure*: When the response occurs, ideally this will be measurable in some fashion so that the scenario is testable.

As an example, in one program, an acquisition quality attribute scenario for "flexibility" could be described by this scenario: "In an environment where the continuing resolution bans new program starts, warfighters in the field demand new capability now. The new capability is added to an existing development contract without change in scope." The scenario elements would then correspond to

Source: Warfighters in the field

Stimulus: … demand a new capability now.

Environment: The continuing resolution bans new program starts

Artifact: … and so an existing development contract

Response: … is modified to add the desired capability

Response measure: … without change in scope

Yet in another program, the same quality attribute of "flexibility" could yield a somewhat different scenario due to the differences between the programs. For example, this other scenario might be: "In an environment where the deployment strategy has not yet been defined, warfighters in the field demand new capability now. The existing development contract is modified to provide an early delivery of the desired capability." Here, the scenario elements are

Source: Warfighters in the field

Stimulus: … demand a new capability now

Environment: The deployment strategy has not yet been fully defined

Artifact: … so the development contract

Response: … is modified to provide the desired capability

Response measure: … to support an early delivery

In both of these cases, the scenarios provide specific meanings for "flexibility" but also indicate how they would imply different acquisition strategies for the two situations.

Working With Acquisition Quality Attribute Scenarios

We expect that the notion of acquisition quality attributes has the same relevance to the acquisition professional that software quality attributes have to the software architect. Thus, in practice, we would hope that a valuable exercise for an acquisition team early in a program's life would be to consider the program they are defining in terms of the acquisition quality attributes implied by the program's business goals (whether explicitly stated or otherwise). In such an exercise, we start with the business goals and, for each one, elicit one or more acquisition quality attributes. These are then used to elicit detailed scenarios that would give precise meaning to the quality attributes. One of the most effective mechanisms for eliciting scenarios is through guided brainstorming where appropriate stakeholders volunteer scenarios that are subsequently reviewed for their importance. One key issue is to explicitly consider the difficulty of satisfying the scenarios. Scenarios that are both of high importance and difficult to satisfy are the ones that will receive the most attention during the subsequent analysis.

Again, working as a group, the scenarios must then be analyzed for consistency with each other. If the consensus is that some scenarios are inconsistent with others, this implies that some business goals are inconsistent with each other and that the program cannot satisfy all of those goals. Ideally, conflicting business goals will be negotiated by the stakeholders until they no longer conflict, thus leading to an acquisition approach that can satisfy the stated goals.

Because acquisition quality scenarios define how the program must respond to events, the scenarios can now be used to determine whether a proposed acquisition strategy will allow the program to respond appropriately to expected future events. The collection of scenarios can be used to shape the program's acquisition strategy so that it will embody the negotiated goals for the program.

Finally, we expect that both the elicitation and analysis of acquisition quality attributes, and their associated scenarios, can be combined with the elicitation and analysis of software quality attributes. In such a case, the consistency analysis can account for both software and acquisition quality attributes, thereby leading to a negotiation between mission and business goals. Although harmony between the two sets of scenarios does not guarantee program success, we posit that if there are conflicts between the software-based and acquisition-based scenarios, it is extremely likely that the architecture will conflict with the acquisition strategy and be a significant detractor to program success.

We are currently in the process of validating our assertion that acquisition quality attributes can be used to define the relationship between business goals and acquisition strategies. We are performing this validation by reviewing various programs, extracting their business goals, developing plausible scenarios, and then defining pieces of the acquisition strategy that would satisfy the scenario.

In addition, we are examining several cases of long-lived programs where we can clearly distinguish between the original goals for the program and changes to those goals. Through interviews, we extracted the business goals and the associated acquisition strategy that the programs adopted, developing acquisition quality attribute scenarios as examples. As new goals were added, we developed new acquisition quality attributes. Our goal with this exercise was not to second-guess or judge the original acquisition strategy but, rather, to show that scenarios can be found to be consistent or in conflict and that the new scenario can be used to judge the effectiveness of the existing acquisition strategy.

Conclusions and Next Steps

Alignment between the architecture and acquisition strategy does not occur naturally. Our research has revealed seven behavioral patterns, or anti-patterns, that lead to misalignments, which have contributed to program restarts, cancellations, and other failures.

We have also learned that the effect of these anti-patterns can most easily be observed through several key entities, and the relationships that should hold between and among them. A key insight derived from this research is that the relationship of the acquisition strategy to the business goals is analogous to the relationship between the software architecture and the mission goals. Our continuing investigations include developing acquisition quality attributes with associated six-part scenarios and validating that they can be used to distinguish between appropriate and inappropriate acquisition strategies.

Our future goal is to create an *alignment method* that assists program office personnel to systematically identify probable areas of mismatch between business and mission goals, acquisition strategy, and software architecture. With such a method, two outcomes would be feasible: (1) program managers can build acquisition strategies that more systematically eliminate one key cause of program failure, and (2) acquisition overseers can better evaluate the adequacy of the acquisition strategy as programs move to different life cycle phases. While this alone will not guarantee program success, we believe that it could be a significant factor in avoiding failures.

References

Acquisition strategy. (2011). In *Glossary of defense acquisition acronyms and terms* (14th ed). Retrieved from Defense Acquisition University website: http://www.dau.mil/pubscats/PubsCats/13th_Edition_Glossary.pdf

Barbacci, M., Ellison, R., Lattanze, A., Stafford, J., Weinstock, C., & Wood, W. (2003). *Quality attribute workshops (QAWs)* (3rd ed.; CMU/SEI-2003-TR-016). Retrieved from Carnegie Mellon

University, Software Engineering Institute website: http://www.sei.com.edu/library/abstracts/reports/03tr016.cfm

Bass, L., Clements, P., & Kazman, R. (2012). *Software architecture in practice* (3rd ed.). Boston, MA: Addison-Wesley.

Brown, W. J., Malveau, R. C., McCormick, H. W., & Mowbray, T. (1998). *Antipatterns*: *Refactoring software, architecture, and projects in crisis.* Hoboken, NJ: Wiley and Sons.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-oriented software architecture volume 1: A system of patterns*. Hoboken, NJ: Wiley and Sons.

Gagliardi, M., Wood, B., Morrow, T., & Klein, J. (2010). *System of systems (SoS) quality attribute specification and architecture evaluation* [SEI webinar]. Retrieved from http://www.sei.cmu.edu/library/abstracts/presentations/20100120webinar.cfm

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software*. Boston, MA: Addison-Wesley.

Maier, M. (2006). System and software architecture reconciliation. *Systems Engineering, 9*(2),146–159.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CA 93943

www.acquisitionresearch.net